

MOVIE RECOMMENDER SYSTEM

*Project Report submitted in partial fulfilment of the
requirements for the Degree of*

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

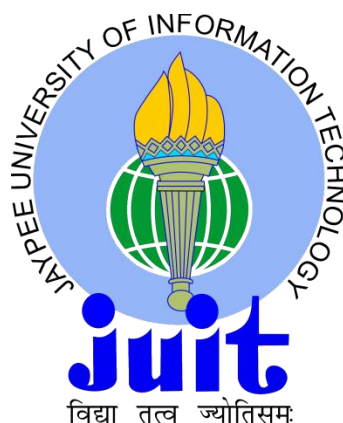
By

AMRITANSHU SINGH

UNDER THE GUIDENCE OF

RUCHI VERMA

Roll No. – 161474



Department of Computer Science and Engineering.

Jaypee University of Information Technology.

Candidate's Declaration

I hereby declare that the work presented in this report entitled “ **Movie Recommendation System**” in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from May 2020 to June 2020 under the supervision of **Dr. Ruchi Verma** .

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Amritanshu

(Signature)

Amritanshu Singh 161474

This is to certify that the above statement made by the candidate is true to the best of my knowledge.



Dr.Ruchi Verma

Computer Science and Information Technology

Dated:

Acknowledgement

The training opportunity I had with **Infosys Private Limited, Mysore** was a great chance for learning and enhancing my technical skills. I consider myself as lucky to have been provided an opportunity by the **Jaypee University of Information Technology** to do training in a leading technology of today. I am also grateful to the **University's Computer Science and Engineering Department** for guiding me and helping to find training as per my need.

I also extend my heartfelt gratitude to the Head of Department of my institute, **Dr. Ruchi Verma** and to all the faculty members of JUIT for providing me with the knowledge that was necessary for me to complete my training efficiently.

I choose this moment to acknowledge her contribution gratefully.

I perceive this opportunity as a big milestone in my career development. I will strive to use gained skills and knowledge in the best possible way, and I will continue to work on their improvement, in order to attain desired career objectives.

Sincerely,

Amritanshu

Amritanshu Singh

Content

- 1. Acknowledgement**
- 2. Student Declaration**
- 3. Introduction**
- 4. About the Industry**
- 5. Python Content**
 - 5.1 History of Python**
 - 5.2 Python Major Version Releases**
 - 5.3 Python-3 Basic Syntax**
 - 5.4 Basic Operators**
 - 5.5 Decision Making & Loops and Control Statements**
 - 5.6 Data Structures in Python**
 - 5.7 Functions**
 - 5.8 Modules**
 - 5.9 Exception Handling**
 - 5.10 File Handling**
 - 5.11 5.13 NumPy**
 - 5.12 Pandas**
 - 5.13 Matplotlib**
 - 5.14 Machine Learning**
- 6. Project- “MOVIE RECOMMENDER SYSTEM”**
- 7. Result**
- 8. References**

Introduction

Training is a great platform during vacations to gain actual field experience for students to learn, to grow and enhance their technical skills.

Python is easy to use, powerful, and versatile programming language, making it a great choice for beginners and experts alike. It leads in the statistical, artificial intelligence, systems tests, to develop Python libraries and applications which address the needs of current and future work in machine learning.

During the training, a complete overview of the technology- “Machine Learning with Python” was provided. The training institute provided with proper study material and daily assignments were given regarding the topic discussed on the particular day. Additionally, an introduction to basic Machine Learning algorithms and tutorial on their implementation was given.

Theories were implemented over various practical coding problems.

.

Introduction of Project:

The title of the project is “**Movie Recommender System**”.

As the name indicates the main motive of the project was to recommend similar movies to user as per their taste and the taste of other people like them. Which is mainly used in Netflix and amazon prime videos. Also used by youtube for recommending us the best video as per our taste.

About the Industry

Infosys Limited is India's pioneer in imparting Certification & Recruitment Company. Set up in 1981, INFOSYS LIMITED started off by providing Certification in Information Security and related Technology Networking, Cloud Computing etc. At the present date the lab also provides you with the latest technology such as Python, Machine Learning, Android, Java etc.

At Infosys the understudies or the applicants are extended to the correct preparing and the activity according to their particular abilities, experience and interests. Additionally, after the consummation of the preparation Infosys Limited stays in contact with the understudy if any assistance is required in the arrangement procedure.

Infosys late spring modern preparing program is intended for the understudies who are hoping to ace their specialized aptitudes. Appin's late spring preparing gives understudies the chance to get hands on involvement with the specialized field. The late spring preparing is a task based preparing program which is comprehensive and spreads the most recent and up and coming advances.

Infosys Limited in Chandigarh appreciates a decent foundation. The office is invested with a helpful learning and information sharing condition. Understudies have advantageous access to fundamental learning instruments and study materials. The workforce group is truly agreeable; consistently quick to help and backing at whatever point an understudy needs that. Every one of them is exceptionally energetic about educating and tutoring. A ton of its past understudies are doing incredible in their expert life, a large number of whom perceive this foundation for that and feel fortunate to have had selected here. This foundation has been fruitful in forming the eventual fate of a significant number of its understudies and keeps on doing as such. It distinguishes the ability in every person, examinations their requirements and supports them in like manner with the goal that they can acquire the required greatness.

As the largest corporate university in the world, the Infosys global education center on the 337 acre campus has 400 instructors and 200 + classrooms at its core, with international benchmarks. Established in 2002, by June 2015, it had graduated around 1,25,000 graduates in engineering. It is a train of 14,000 employees on different technologies at a given point of time.

Abstract

In the dissemination of content, it soon becomes a very interesting problem how to locate one's favorite film among a huge number of films. In particular when the user does not have clear Target-film information, personalized recommendation system can play an important role. In this paper we develop and incorporate a prototype of a film recommendation program through research of KNN algorithms and collaborative filtering algorithms in conjunction with the current one Recommendation needs for films. We then give model design for the Clear Theory and JAVAEE relational database frameworks. Finally the test results showed the system has a good effect on recommendations.

Litrature Survey

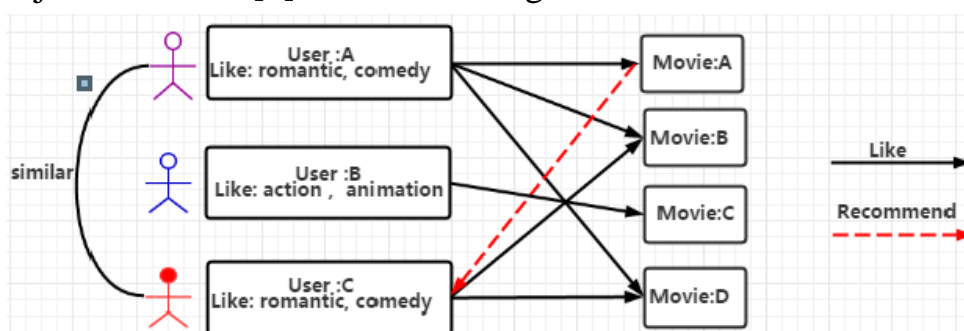
With Internet technology rapidly evolving[1], today's Company entered the Web 2 era, with information overload Make reality. How to find the information you need inside Mass of data became a hot topic of research. Movie is amongst others The main spiritual fun, too, has the problem of Overloading Information. To solve that problem, this Paper bringing out a design film idea Set of recommendations[1,2].

Try personalized recommendation Consumer tastes and attributes through collecting and Study of past actions in order to learn what sort of individual the Customer is, What sort of action the customer likes, what form of behaviour Types of things that the user likes to share, and so on[3.4.5], and Finally, grasp what consumer functions and expectations are Centered on the framework law, and suggested Information and goods of interest to the User[6.7]. Customized recommendation system is something of a Technology to filter information. It is a build in device That is a combination of different data mining algorithms And details specific to the customer to satisfy the needs or ability User desires. The common System of recommendations is Categorized as a recommendation framework focused on the content, Collaborative recommendation filtering system, and hybrid Recommended Unit[9,10]. All Recommendations Algorithm has different range of uses and conditions of use, which results in Use of various recommendation algorithms for the Same recommendation for information.

In the application in question The program is usually a variant of the suggestion framework Program of suggestion. That is, to combine the benefit of Every algorithm suggested for the specified method To effectively improve the effect of the recommendation.

Collaborative Filtering Algorithm

Collaborative filtering algorithms are categorized as user based filtering Collaborative filtering algorithms[4] and community dependent filtering algorithms Filtering collaboratively. The fundamental principles of both are quite right Similar, and mainly introduces the user-based section Algorithm recommended for collaborative filtering. Bottom line The idea of a collaborative filtering algorithm is to Insert similar-interest user information to object Benutzers[7]. As shown in figure.



User A prefers films A, B , C and consumer C prefers films B, D,Therefore, we may infer the user A and user C tastesThey are very similar. Since user A also loves film D, so do we Could infer that user A might also love item D, so item The user would be advised to D. The basic idea behind the Algorithm is based on user's history score records. Check Neighbor consumer as u 'who has identical goal value User u, then recommend neighbor's items User u 'loved targeting user u, predicts which target score The user u may give a score on the item Neighbor user calculation u 'on item. This algorithm is Consists of three basic steps: Calculation of user similarity, Calculation of nearest neighbor selection and score prediction.

Python Content

5.1 History of Python:

Python was developed in 1980 by **Guido van Rossum** at the National Research Institute for Mathematics and Computer Science in the Netherlands as successor of ABC language. The name was inspired from a TV show aired on BBC channel, “Monty Python’s Flying Circus”.

5.2 Python 3- Basic Syntax

- Python identifiers
- Lines and Indentation
- Comments
- Data Types:
number(int,float,complex),bool,string,bytes,bytearray,range,list,tuple,set,dictionary,none.

5.3 Basic Operators:

Types of operators:

- Arithmetic: +, -, *, /, %, **(exponent), //(floor division)

- Comparison: ==, !=, >, <, >=, <=
- Assignment: =, +=, -=, *=, /=, %=, **=, //=
- Bitwise: &, |, ^, ~, <<, >>
- Logical: and, or, not
- Membership: in, not in
- Identity: is, is not

5.4 Decision Making & loops and control statements:

- IF... ELIF...ELSE... Statements
- While
- For
- range() function usually used along with the for loop as a counter.
- Loop control statements: break, continue, pass(a null operation)

5.5 Data Structures in Python:

- Lists: ordered collection of data.
- Dictionary: mappings between a unique key and a value pair.
- Tuples: used to present things that shouldn't change.
- Sets: contains unique and unordered elements.

5.6 Functions:

- Syntax
- Function Arguments: Required, Keyword, Default, Variable length
- Recursive Functions
- Anonymous Functions
- Nested Functions
- Decorators
- Lambda function

5.7 Modules:

- Import statement
- The from...import statement
- Creating Modules
- Reloading modules

5.8 Exceptions Handling:

Python provides two very important features to handle any unexpected error in Python programs and to add debugging capabilities – Exception handling and Assertions. exception handling in Python like other

Languages Python also gives runtime errors which can be handled by our exception handling with the help of try except some of the standard exceptions which are most frequently included our index error import error input output error 0 division error and type error you can check the exception hierarchy from over here here we can see all the built-in exceptions let us try to access the array elements whose index is out of bounds and handle the corresponding exception here we have an array containing elements till the second index and we are trying to access the value at the third index but accessing is a part of try as soon as we try to access the fourth element or the third index then we move then we move to the except statement let's run this code we can see that an error must have occurred and it is handled if the value of a is greater than foo name error is caused if a equal to three zero division error is caused this can handle both the errors let's run this code we can see that the error occurred is handled in Python you can use else Clause on try except block which must be present after all the except clauses the code enters the else block only if try Clause does not raise any exception like over here for the value to 3 the code enters the else block because it does not cause 0 division error the Rays treatment allows the programmer to force a specific exception the sole argument and raise indicates the exception to

be raised this must be either an exception instance or an exception class that is a class that derives from the class exception like here we are raising a named error as a string as at a very simple string.

- Try...except
- Try...except...else
- Try...except...finally
- Nested try...except...finally
- Raising exceptions
- Customized/User defined exceptions

5.9 File Handling:

Python too supports file handling and allows user to handle files i.e. to read and write files, along with many other file handling options, to operate on files.

Let's have a look at Python file handling in this course. On neural networks we're

going to deal with essentially three types of files, we're going to deal with

CSV files, image files, and text files. CSV files usually have the .csv extension, you

can think of those as looking like Microsoft Excel files. In fact on most computers you double-click a CSV file and Excel will pop open. Image files, we'll

deal mostly with PNG or JPEG there's also text files and others. Images are one

of the things that neural networks do particularly well so we'll see a lot of different images in this in this class. And then text files have the .txt extension. Those are usually just raw text and that has to do often with natural language processing other types of files that we'll see. Briefly in this class are JSON and h5 these files will come from three primary locations your hard drive. Now this is important if

you're using Windows you'll see paths like this, if you're using a Mac you'll see files like this, most of you I assume will be using Google CoLab so watch through the tutorial on how to read those from Google CoLab you'll basically have a path just like this, like this, it'll be close to Macintosh. CSV files can be read with Pandas. We will deal with Pandas a lot in the next class. Another thing to note here that is the way that i'm doing this. I am giving you web addresses for most of the csv files that will load data HeatonResearch.com that's my own URL which is unique to this class and I have all the data files there. This way if you're running in Google collab or Mac or Windows or whatever it will work just fine. The above command loads Fisher's iris dataset straight from the Internet. Ok, it's loaded you can display the first five records of that. And this is a very classic dataset if you haven't seen it before it is basically four measurements from flowers and it defines the species. Here there's actually three different species at the very beginning since the file is sorted you're only seeing the first iris which is sat Setosa. This is a classic classification dataset where you try to use these values to classify what type of iris flower it is. Reading an image requires the PIL library but it works similar to reading CSV files you're basically going to do this and run it and it will load this image from from this URL. And this is one of the buildings from Washington University, it doesn't actually look like that anymore thanks to all the construction but for at least the grass in front of it. But this is this is a way that you load a JPG. Now when we do some of these exercises we're going to have lots and lots and lots of images. You may literally harvest the images this way but you'll want to put them on a folder on your Google Drive. What's good about streaming them is you don't actually load the whole CSV file into memory at once. if you're just reading a CSV file and calculating statistics on each row you really don't need to load the whole thing into memory you can simply read each row and process it. So here I am taking the iris data set

I am opening it up for streaming that means the whole thing is not loaded into memory at once and I'm going to sum each of those four values that we saw in there. I create an array of four zeros and NumPy and I keep a count of how many rows I've read and then I run this and you'll see this is this is essentially the average of each of those four values I am going to loop over the lines so each each line I basically kept to I convert that line into. Let's see more of this when we when we deal with NumPy but this is how you create a numpy array I'm taking locations 0 to 4 only. I don't want the species the species is the fifth and I convert them into floating-point because they're integers and because they could potentially come in as integers or strings or other values. I want them as floating points if the line is the right length I skip any empty lines. I believe there's an empty line at the end I sum it up this is a vector summation so this is basically taking everything that's in that line too so the array of those four values and I'm adding it to that other array that I have and I keep the count. Incremented so that's a vector addition that's adding for our numbers just like this is a vector division this returns a vector or an array back but it divides each element by count vector mathematics is very useful in this this course, because we're often dealing with vectors which are a low dimension form of a tensor. So we we deal anytime that we can perform math across an entire tensor at once, it's a great optimization really a text file this is the United States Declaration of the independence this is just a text file that I was able to find that had a URL, and it was a pure text file you can run it and it basically prints out the raw text of the Declaration of Independence of the United States of America. Thank you for watching this video this concludes file handling and Python in the next video. We're going to look at functions lambdas

and MapReduce this content change is often so subscribe to the channel to stay up to date on this course and other topics and artificial intelligence.

- Modes: r,w,a,x,r+,w+,a+
- Pickling,unpickling using Pickle Module
- Reading and writing Binary files

5.10 NumPy

NumPy stands for “Numerical Python”. It is a library consisting of multidimensional array objects and a collection of routines for processing of array.

NumPy stands for Numeric Python and is a library that can be used for scientific computing. A fundamental part of the NumPy library is the array object capable of holding elements in multidimensions and performing element wise calculations over the whole array. In addition, the library includes tools for linear algebra such as matrix and vector products, and matrix solving and inverting functions. The NumPy library’s design of broadcasting functions are versatile and help determine how the library treats arrays with varying shapes during calculations.

Because of these features unique to the library, NumPy has become an essential tool for scientific computing. The NumPy library can be installed using pip or a Python distribution such as Anaconda.

A Python distribution such as Anaconda includes NumPy along with many other common scientific packages such as SciPy, Pandas, and Matplotlib. I will demonstrate installing NumPy using pip, but if you are interested in installing the Anaconda Python distribution, check out our video using Anaconda for installing scientific packages. When installing a new library in Python using pip, I recommend first creating a virtual

environment to keep the scope of installed libraries local to the project. This is an alternative to installing libraries globally for a system user. The virtual environment that I use is called venv, which is included in the Python standard library, although many options for creating virtual environments in Python exist. Now we will setup a virtual environment using venv by typing `py` (or `python3` on MAC) then add the module option `-m` followed by `venv`.

Next specify the directory where you would like to place the virtual environment. I will call my directory `NumPyVenv`. If this directory does not yet exist, it will be created automatically. When this command is executed, the virtual environment will be created and within it a `pyvenv` configuration file is setup. To activate the new virtual environment use one of the following commands depending on your operating system.

Because I am on windows, I will use the command `NumPyVenv\Scripts\activate.bat`. The name in the parentheses on the left side tells us we are within the activated virtual environment.

Now that the virtual environment is running, we can install NumPy using the command `pip install numpy`.

The NumPy library has now been installed in our virtual environment and we can start using the library.

5.11 Pandas

pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. It offers data structures and operations for manipulating numerical tables and time series, which is a Panel Data. Therefore, the library is named is Pandas.

with pandas you can load prepare manipulate model and analyze data you can join data you can merge data you can reshape data you can take data from different data bases and put it together and analyze it you can do pretty much anything you want to with data and it all revolves around a structure called a data frame let's take a look at some examples I'm gonna show you how to use pandas very briefly on this Titanic LS

file so this is a file an excel file that shows the passengers on board the Titanic it has what class they were in whether or not they survived their names their sex and age and so on so there's a lot of information about the passengers it's the entire passenger list and there are just over 1,300 entries and this is quite a famous data set so let's explore

it we're gonna go to about Jupiter notebook and we're going to import numpy as NP we're going to import pandas which is what we're interested in here as PD so we've created a variable called Titanic DF this read XL is a panda's function and there's a read CSV and other functions that will read different file types we will run these two cells and then using this method here we can at the data that we've got so we've now created a data frame and this is a really important structure in pandas and when you learn more about pandas you'll learn all about data frames and these are the data frames that you can merge and join and do so all sorts of things with it's a really really useful tool and so we can see here that we've got the first five records of this data set and the next thing we can do is we can describe the data set so it'll tell us the count we can see we've got 1046 age records and so a little short of the full amount we get the minimum age and the maximum age and then the quartiles there so that's quite useful and if you look at the affairs that's quite interesting so the mean Fair was thirty-three pounds I guess but the maximum fare was 512 now using this drop command we are going to get rid of some of the data from our data frame because it's not going to be that relevant so we're going to get rid of the ticket column the cabbing column the boat column and the body column and then we're going to have a look at what's left so let's have a look at that we now have a new data frame with less information but it's more relevant information so we've we've trimmed the data frame a little let's carry on so we're going to have a look now at doing a plot so what I'm going to do is I'm going to use this value count function on our data frame but I'm going to do it just on the survived column and then I'm going to plot it using a bar plot let's run that and there we have the results of that plot very quickly you can see that where we have a zero those are the people that died and where we have a one of those are the survivors so data visualization with pan is it's very quick indeed let's have a look now at the proportion of people that survived let's get that a figure so we run the mean command on the survived column and we get 38 percent so you can see with

pandas there are a lot of tools that allow you to do statistical investigation into your data very easily indeed now what we're gonna do now is we're gonna group the data in a different way I want to group it by the sex of the passenger to see how that affected the outcome okay so what we have here is we have everything now grouped by male and female I also want to see whether the class of the traveler played a role in their likelihood of survival so now we do group by but this time we do it on sex and the class and we're gonna get the mean figures of both so let's have a look so now you can see we've broken this down into female male and then the class of travel and this is really revealing isn't it so females in first class had a 97 percent chance of survival whereas men in third class or males in third class had a 15% chance of survival and finally let's see what effect age played because they said didn't they women and children first so perhaps we can see whether that was true so to do that we do the same command as before but this time we do it only for ages under 18 and these are the results so in first class those under 18 eighty seven and a half percent of them survived in second class all of them survived out of the females and in third class 54% of females under the age of 18 survived if we look at the males in first class eighty-six percent of male survived in first class in second class it was 73 percent but in third class it was only 23 percent so in a very few lines of code we've managed to really examine our data very well indeed and that's really one of the strengths of pandas you can do so much with so few commands another strength of pandas is working with time series and it's used a lot for this in academia I want to show you an example now of pandas using time series with an example from the stock market this is a short example just to show you a few of the things that pandas can do with the time series but obviously it can do so much more than this we're going to have a look at some stock price data so we've got Apple Microsoft and I've used Quon Ville to import the data that's already loaded so let's just have a look at the header of Microsoft the share price just to see what we've got here okay so we can see the first five entries the data goes back to 1986 we've got open high/low close we've got the volume we've got the ex-dividend information of the split ratio we've got the adjusted prices as well so we've got a lot of information there now I just want to plot the adjusted closed price from Microsoft so let's have a look at that and see how we do that as you can see it's very easy we just take the data frame

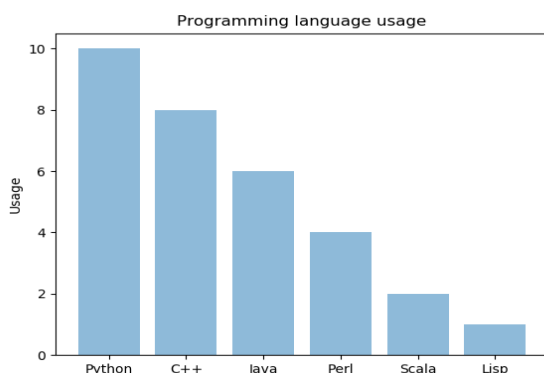
name we choose the column name that we want and we just type plot next to it and if we do that we get a nice graph going back to 1986 up to the present and showing us the price of Microsoft okay now let's have a look at the index of Microsoft because this is the key bit if we type `ms dot index` that tells us that the index is a date time index so this data has been loaded in to our data frame with the index being the date and that's really useful and what that means is we can do some very interesting things so for example if we just wanted to see the price of Microsoft in 2018 we can just choose 2018 just like that and pandas says the rest for us so let's have a look and that's the price in 2018 if we just want to have a look at the price in March in 2018 then we just put in 2018 slash oh three and pandas does the


work for us and there it is that's the price in March and if we wanted to do a range say from the beginning of 2018 to the end of March again we just put in the range that we want we put the column that we want to plot on the type plot and pandas does that for us to what I want to do now I want to combine both stocks into one data frame so I can see that information plotted together and here I'm going to join m/s price with

Apple price and if we run that now we have this data frame and now we just want to plot it and there you have the plot of the two stocks okay now what if we wanted to just look at what happened to the price in 2017 for example well we do what we did before we can have a look just at the 2017 data and then what other things might we want to do well you can do a rolling average let's have a look at the rolling average and there it is and if you wanted to do a rolling standard deviation to see how much the stock price is move on a daily basis when you can do that too and you can see there straight away that Apple it's a little more volatile than Microsoft this example really is just to show you how powerful pandas is now that should give you some idea of the capabilities of pandas.

5.12 Matplotlib

Matplotlib is a python library used to create 2D graphs and plots (histograms, power spectra, bar charts, errorcharts, scatterplots, etc.) by using python scripts. It has a module named `pyplot` which makes things easy for plotting by providing feature to control line styles, font properties, formatting axes etc.



 barchart.py - F:/Training_Python_ML/Project/Project2/fig/barchart.py (3.7.4)

File Edit Format Run Options Window Help

```
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.pyplot as plt

objects = ('Python', 'C++', 'Java', 'Perl', 'Scala', 'Lisp')
y_pos = np.arange(len(objects))
performance = [10,8,6,4,2,1]

plt.bar(y_pos, performance, align='center', alpha=0.5)
plt.xticks(y_pos, objects)
plt.ylabel('Usage')
plt.title('Programming language usage')

plt.show()
```

5.13 Machine Learning

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. **Machine learning focuses on the development of computer programs** that can access data and use it learn for themselves.

5.13.1 Types of Data:

- Structured data
- Semi Structured data
- Unstructured data

5.13.2 Learning Classification

- Supervised
- Unsupervised

5.13.3 Algorithms

- KNN
- K-Means Clustering
- Logical Regression
- Decision Trees

Project- “Movie Recommendation System”

6.1 Introduction:

Recommender systems we're going to start with an Overview of recommendation systems and why they are necessary then we are going to look at the two most common types of recommender systems content based systems and collaborative filtering and finally we can look at how to evaluate recommender systems to make sure they're in a good job let's start with an overview imagine any situation where a user interacts with a really large catalog of items now these items could be products at Amazon they could be movies at Netflix they could be music from Pandora's catalogue or it could be you know news items on Google News what really matters is that there are tens of thousands or hundreds of thousands or millions of items a really large catalog and the user is interacting with this catalog now there's two ways in which a user can interact with a large catalog of items the first is search the user knows what they are looking for and they go and they search the catalog for the precise item that they are looking for now when you have a really large catalog of items very often the user doesn't know exactly what they are looking for and this is where recommendations come in the system recommends to the user certain items that they think the user will be interested in based on what they know about the user now why do we really need such recommendations the key that made recommendations so important and a why recommendation system develops so much in the last 10 or 20 years is that we moved from an era of scarcity to an era of abundance.

What do I mean by this imagine that you are out shopping 20 years ago and you'd go to a local retailer and you'll find a certain number of products on the shelves of the local retailer now even in a very large retailer like like a Walmart for instance shelf space is a key it's a scarce commodity it limits the number of items that a retailer can carry shelf space is expensive because it involves real estate costs and therefore a retailer can carry only a certain number of products now similar situation applies in the case for example of TV networks a TV

network can carry only so many shows because there's only so many number of hours in a day and there are only so many movie theaters so they can only serve the screen a certain number of movies now once the internet was developed things changed the web enables zero-cost disseminate of information about products and what this means is that we can have many more products than ever before there is no shelf space limitation on the number of products.

That's why the number of products on Amazon is much much more than the number of products available at any physical retailer the number of you know movies available on Netflix is more than the number of movies that were ever away available at a blockbuster and so on this new zero cost dissemination of information gives rise to a phenomenon that's called the longtail phenomenon.

Let's examine what this is now imagine a graph where on the x - axis we have taken the items in the catalog remember items might be books or music or video or news articles and you've ranked these items by popularity so the most popular items are on the left and as you move towards the right the items become less than less popular what do I mean by popular well I mean the number of times the item is purchased in a week or it could be the number of times a movie is viewed in a week or a month or some some fixed time period now on the y axis we have the actual popularity which in this case I've shown as number of purchases per week it could be number of views per week or it could be number of you know plays per month for for music and so on so in general you have items ranked by popularity along the x-axis and the popularity itself along the y - axis now when you take items you know in a large catalog and you rank them and you plot them on this curve you get a curve that looks like this you can see that this curve you know has a very steep fall initially the the you know you have a really really a few really really popular items and then as you move towards the right as the you know as the item rank becomes greater the popularity falls off very steeply but at a certain point you can see that this popularity stops you know you know falls off less and less deeply and you know it never quite reaches the x axis the interesting thing here is that there is a cut-off point beyond you know items that are less popular than this cutoff point you know might be purchase perhaps just to once a week or maybe once a month if you're a physical retailer like a Walmart it's not economic to stock the

system because the rent cost of stocking the item is more than you make buy when you sell the item and therefore a retailer any right-thinking retailer does not stock items that are unpopular that you know they only stock the head of this distribution so there's this cutoff point that I show on this graph here and items that are more popular than this the the more popular items are available at a retail store but the less popular items the items are to the right of the cutoff point are not available at any retail store they are only available online now this phenomenon applies to books to music to movie to videos to news articles.

for example,

There are only so many news articles in newspaper but when you go online you can see the rest of the news articles the less popular news articles that are off to the right the piece of the curve that is to the the piece of the curve here that is to the right of this dividing line is called the long tail these are the items that are available only online the interesting thing is the is this area under the curve here and you can see the area under the curve here is quite significant in fact in some cases area under the curve on the right is about as large or could be even larger than the area of the curve under the curve on the on the left so you have all these items that could now be found in a physical store but there can be only found online but there are so many of them that it's very hard for any user to find all these items right so when you have this area of abundance and you have so many items and many of them are only found online how you know how do you introduce a user to all these new items that they may not otherwise find when you have more choice like this when you have these millions and millions of items that are only available online you need a better way for the user to find all these items the user doesn't even know where to start looking and that's where recommendation engines come in so recommendation engines book in the case of many many kinds of items books music movies news articles interestingly they even work in the case of people.

for example,

When you go to Facebook or LinkedIn or Twitter there are so many people that you do know who to follow a good friend and so Facebook or LinkedIn or Twitter makes recommendations to you on the people that you know that you

could follow a friend I led this point with an interesting anecdote that shows you the power of recommendation engines a several years ago a book was published called touching the void it's a book about molten earring it's a very very good book the book came out it didn't make much of a ripple you know a few people bought the book it got some decent reviews but it never became a best-seller and then a few years after touching the void a new book was published on mountaineering called into thin air now into the net effect of traction and lots of people started buying into thin air Amazon noticed that a few of the people who bought into thin air had also bought touching the void so they started recommending touching the void to people who bought into thin air and lo and behold those people started buying touching the wide as well the interesting point is this may touching the void a best-seller in fact it became a bigger bestseller even than into thin air even though a few years ago a debt sank without a trace so this example should show you the power of recommendation systems there are these items these sort of gems like touching the void know that people don't know because they don't know to look for them but a good recommendation system can expose people to these hidden gems that they wouldn't have known about otherwise so let's look at types of recommendation systems the simplest and the oldest kind of recommendation is editorial or hand curated you might find a list of favorite for example when you go into your favorite neighborhood book store you might find staff picks certain books are marked off as topics right and these are editorial our hand curated and on certain websites you'll see a list of staff favorites or you'll see a list of essential items these are essentially built by hand and another place where you will see these editorial recommendations is often on the home pages of websites for example if you go to the the home page of most popular websites including product websites you'll see editorial pick these are products that have been picked by the rhetorical staff too feature on the home page the drawback with editorial or hand curated recommendations is that it's done entirely by you know by the staff of the website and there is no input from the users of the site so when you go beyond editorial recommendations the next simple thing that you can do is simple aggregates on many websites you'll see a list of top ten or most popular or most recent.

for example,

If you go to youtube you can see the most popular videos for instance right so these are simple aggregates which sort of take into account user activity to make recommendations to other users but these recommendations don't depend on the user they only depend on you know the aggregate activity of a lot of other users the third and most interesting kind of recommendation to us is recommendations that are tailored to individual users right for example book recommendations tailored to your tastes or movie recommendation based on the movies that you watch previously or music recommendation based on your music interests and this is our focus here recommendations that are tailored to individual users so let's look at a formal model let C be a set of customers and S a set of items we go to create a function called a utility function or a utility matrix the utility function is a function that looks at every pair of customer and item and maps it to a rating okay our R in this case is a set of ratings and for example R could be a star rating from 1 star to 5 star or R could be a number between 0 and 10 in general R is a totally ordered set so that you know a lower value indicates that the user liked the product less and a higher value indicates that the user liked the product more let's look at an example of a utility matrix now on the top we have 4 movies here but are a lot of the rings' matrix and Pirates of the Caribbean and down here we have 4 users a lesbo Carolyn David and the utility matrix gives you ratings for certain movies and certain users for example Alice has rated avatar and matrix but not Lord of the Rings or Pirates of the Caribbean whereas carol has rated you know has rated the same two movies Bob has rated a lot of the Rings and Pirates but hasn't rated avatar no or matrix now it could be that these users have not seen these movies or it could be that they have seen the movies but not bother to rate them so in general utility matrix like this is going to be sparse you know most of the users haven't seen most of the movies and there are going to be values in some of the you know some of the locations the key problem in recommendation systems is to figure out these unknown values for example you've seen that Alice has rated avatar and matrix but hasn't rated Lord of the Rings so the question is can we figure out what Alice's rating for a lot of the Rings will be based on her other ratings can you figure out whether she'd like pirates or not right so this is the key problem for recommender systems once we find out for each user certain movies that they would have rated highly or with the system thinks they might have rated highly then we can recommend those movies to those users so there are three key problems in the space of recommender systems the first is gathering the known ratings ratings in the matrix now in the previous slide

when you look at the utility matrix it was already filled in with certain values but how do you get a gather those values in the first place so that's the key that the first problem that you need to tackle the second problem is to extrapolate unknown rating from known ratings but we are mainly interested in the high unknown ratings we're interested in those ratings where a user would have given a high rating to a movie we're not interested in the average or the low ratings because they never going to recommend those movies to the user and finally the third key problem is evaluate extrapolation methods once you have a recommendation system that can extrapolate unknown ratings from known ratings how do you know that the recommender system is doing well this very except you know the evaluation methodologies come in let's start with the first problem that of gathering datings the first and simplest way of gathering ratings is is what what I will call explicit method simply ask people to rate items now this method is good because the you know you're asking people to directly rate items and you're going to get you know but and you can decide on what scale people are going to rate item for example you can say you can ask for ratings on a one to five star scale or you can ask people to rate on a scale from zero to ten or you can just ask people to say whether they liked an item or did not like it so the explicit method has the advantage of simplicity and of getting direct responses from users the problem though is that it doesn't scale only a small fraction of users who viewed a movie or listen to a you know piece of music or bought a product actually bother to leave a rating or review most users don't actually leave ratings or reviews so while the data that explicitly gathered excellent data it it's not sufficient in most cases for recommendations because only a small fraction of users actually leave ratings and reviews since explicit ratings don't scale lot of sites use implicit ratings now the idea behind implicit ratings is to learn ratings from other user action.

For example an online shopping website might have a rule that a purchase implies a high rating now the nice thing about implicit ratings is that they're much more scalable than explicit ratings because the user doesn't have to explicitly rate an item and there are way more other actions such as purchases then there are ratings the problem though is that it's very hard using implicit ratings to learn low ratings it's quite easy to learn high ratings because you might have a rule that purchase implies a high rating but you can never learn a rating that a user disliked a product implicitly in practice most recommender systems and most websites use a combination of explicit or implicit ratings where explicitly ratings are available they use them but they supplement them

with implicit ratings when needed let's move on now to the central problem of extrapolating utilities or extrapolating unknown utilities from known utility values the key or central problem that we have to surmount to extrapolate utilities where the matrix U the utility matrix is very very sparse most people have not related most items and this introduces a slew of problems that will.

Come across shortly the second problem we have is a cold start problem when you have a new item or a new user the new item doesn't have any ratings and new users have no history so this is known as a cold start problem and we had a tackle this problem as well in due course there are three approaches to building recommendation systems the first is content-based recommendations the second is collaborative filtering and the third is latent factor based models let's start with content-based based approaches

In 2009 Netflix has held a competition where they gave out the prize money of 1 million dollars for only 10% of improvement in their existing recommendation algorithm although sources suggest that they never got into production but the concepts and techniques that were developed during this competition are still considered to be biggest leaps in this field the recommendation systems as we know today I usually divided into two categories the First of all, the content-driven recommendation mechanisms depend on the characteristics and qualities of the object itself, and for example, if you decide to suggest films to a customer, something you will do is look at films they enjoyed in the past and consider related films depending on the producer of the project, the actors that were in the project tags are qualities certain critics We could use recommendation engine ratings given by other users.

Recommendation systems are basically of two types :

1. Content based Filtering

The main idea behind content-based recommendation systems is to recommend items to a customer X similar to previous items rated highly by the same customer for example in example of movies you might recommend movies with

the same actor or actress directors genre and so on in the case of websites ,blogs or news we might recommend articles with similar content or on the simple similar topics in the case of people recommendations we might recommend people with many common friends to each other so here's a plan of action we're going to start with the user and find out a set of items the user likes using both explicit and implicit data for example we might look at the items that the user has rated highly and the set of items a user has purchased and for each of each of those items we are going to build an item profile an item profile is a description of the item for example in this case we are dealing with geometric shapes and let's say the user likes a red circle and a red triangle we might build item profiles that say that the user likes red items right or they order a user like circles for instance and from these items from these item profiles we could infer a user profile the user profile in first the likes of the user from the profile has items the user likes because the user here likes a red circle and a red triangle we will further the user likes the colour red they like circles and the like triangles now once we have a profile of the user we can then match that against the catalogue and recommend other items to the user so let's say the catalogue has a bunch of items in it some of those items are red so we can recommend those to the user so let's look at how to build these item profiles for each item we want to create an item profile which we can then use to build user profiles so the profile is a set of features about the item in the case of movies for instance the item profile might include author title actor director and so on in the case of image and videos we might use metadata i.e details about data and tags in the case of people the item profile might be a set of friends of the user even though the item profile is a set of features it's often convenient to think of it as a vector the vector could be either boolean (0 or 1) or real valued and there's one entry per feature for example in the case of movies the vector might be the item profile might be a boolean vector and there is a 0 or a 1 for each actor director and so on depending on whether that actor or that director actually participated in that movie let's look at a special case of text for example you might be recommending news articles now what's the item profile in this case the simplify term profile in this case is to pick the set of important words in the document or the item how do you pick the important words in the item the usual heuristic that we get from text mining is a technique called tf - idf or term.

TF – IDF(Term frequency Inverse document frequency)

$$TF_{ij} = \frac{f_{ij}}{\max_k f_{kj}}$$

$$IDF_i = \log \frac{N}{n_i}$$

where,

n_i = number of documents that mention term i

N = total number of docs

Note: we normalize Tf for longer documents.

For example let's say the feature is a certain bird or the word Apple and in the document that we're looking at the word Apple five times but there's another document where the bird Apple appears 23 times and this is the maximum number of times the word Apple appears in any document at all then the term frequency TF_{ij} is five divided by twenty three now I'm glossing over the fact that we need to normalize DF to account for the fact that document lengths are different let's just ignore that for the moment now the term frequency captures the number of times a term appears in a document intuitively the more often a term appears in a document the more important a feature it is for example if a document mentions the word Apple five times the word Apple is more important in that document than another document that just mentions it once but how do you compare the beat of different terms for example you know the red bird appearing just a couple of times might be more important than a more common word like the appearing thousands of times this is where the document frequency comes in let n_i be the number of documents have mentioned the term i and let n be the total number of documents in the whole system the inverse document frequency for the term i is obtained by dividing n by n_i the number of documents that mentioned

the term A_I and then taking the logarithm of that of that fraction notice that the more common term the larger and I and the larger and I the lower the IDF the IDF function ensures you know gives a lower weight to more common words and a high of 8 to rarer words so people these two pieces together the tf-idf score of feature I for document J is obtained by multiplying the term frequency and the IDF so given a document you compute the tf-idf scores for every term in the document and then you sought all the terms in the document d_{fid} of scores and then you have some kind of threshold or you or you might pick the set of birds as the highest tf-idf scores in the document together with this course and that would be the top profile so in this case the dog profile is a real-valued vector as opposed to a boolean vector now that you have item profiles our next task is to construct user profiles let's say we have a user who's rated items with profiles I_1 through I_n now remember I_1 through I_n are are vectors of of entries so let's say this I_1 plus I_2 I_3 and so on and here is I_n these are each is a vector in a high dimensional space with many many entries now the simplest way to construct a user profile from a set of item profiles is just to average the item profiles your n is the total number of item profiles so if I take all the item profiles in the users you know the of all the items the user has has rated and then take that average that would be a simplest way of constructing a user profile now this doesn't take into account that the user liked certain items more than others so in that case we might want to use a weighted average where the weight is equal to the rating given by the user 4 for each item then you would have a weighted average item profile a variant of this is to normalize these weights using the average rating of the user and we'll see an example that makes this idea clear and of course much more sophisticated aggregations are possible here we only looking at some very simple examples let's look at an example that you know that will clarify weighted average item profiles and how to normalize weights.

Here for example,

→ Boolean utility matrix:

what's a boolean utility matrix all we have is information of whether a user purchase an item or not for example so each entry is either a 0 or a 1 let's say the items are movies and the only feature is actor the item profile in this case is a vector with 0 or 1 for each factor 0 if then that actor did

not appear in that movie and 1 if that actual appear in that movie suppose user X has watched 5 movies and 2 of those movies feature actor a and three of those movies feature actor B now the simplest user profile is just the mean of the item profiles now remember there are 5 vectors and two of those have a 1 a 4 feature a and so the weight of feature a is going to be 2 divided by the total number of item profiles which is 5 which is 0.4 and the weight of feature b correspondingly is going to be 3 by 5 let's look at a more complex.

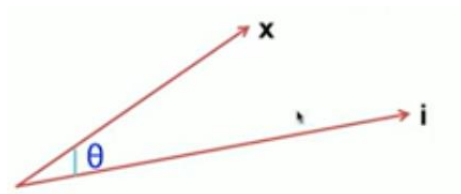
→ star ratings:

have star ratings in the range 1 to 5 and the user has once again watch 5 movies and there are two movie starring actor a and three movie starring actor B the movies that occur a starred in the user rated 3 and 5 whereas a movie that they're active be acted in the user rated one two and four since we have 5 star ratings and the user gives lower ratings for movies they didn't like and higher rating for movies they liked it's somewhat apparent from these ratings that the user like at least one of the movies from from actor a and one of the movies from actor B but didn't he but they really didn't like to of actor B's movie so once that were rated one and two one and two are in fact negative ratings are not positive ratings and we'd like to capture this fact the idea of normalizing ratings helps us capture the idea that some ratings are actually negative ratings and some appositive ratings but the baseline you know users are very different from each other some users are just more generous in their ratings than others so for user a for instance a for might be a widely positive rating whereas for another user for my despi an average rating to sort of capture this idea we're going to baseline each users ratings by their average rating so in this case the this users average rating is a three if you average all the five ratings that the user has provided the average rating is a three and so what you're going to do is to subtract the average rating from each of the individual movie ratings so in this case the movies with actor a the normalized ratings in that case instead of three and five becomes zero and plus two and for actor B the normalized ratings become minus two minus one and plus one notice that this captures intuition that the user did not like the the first two movies with actor B whereas he really liked the the second movie with actor A whether the first movie with actor A was we know was kind of an average movie once you do

this normalization then you can compute the profile profile weights but in this case we divide not by the total number of movies but by the total number of movies with a specific feature so in this case there are two movies with actor a and profile wait for actor a the the feature actor a is 0 plus 2 divided by 2 which is 1 and similarly the feature actor B as a profile weight of minus 2 by 3 this indicates a mild positive preference for actor A.

Mathematics for making predictions:

$$U(x,i) = \cos(\theta) = (\mathbf{x} \cdot \mathbf{i}) / (|\mathbf{x}| |\mathbf{i}|)$$



Lower the angle theta more close the two objects are to each other , now that you have user profiles and item profiles the next task is to recommend certain items to the user the key step in this is to take a pair of user profile and item profile and figure out what the rating for that user and item pair is likely to be remember that both the user profile and the item profile are vectors in a high dimensional space in this case I've shown them in a two dimensional space when the reality of course they are embedded in a much higher dimensional space.

When you have vectors in a higher dimensional space a good distance metric between the pair of vectors is the angle theta between the pair of vectors in particular you can estimate the angle using the cosine formula T theta the angle between the two vectors is given by the dot product of the two vectors separated by the magnitudes function and this distance U in this case we are going to.

The angle theta and the similarity of the cosine is the angle 180 minus theta, the smaller the angle the more similar the item X and the more

similar the user X and the item I are 180 minus θ is going to be larger but for convenience we are going to actually use the cosine of θ as our similarity measure notice that as the angle θ becomes smaller $\cos \theta$ becomes larger and as the angle θ becomes larger and larger the cosine becomes smaller and smaller in fact as θ becomes greater than 90 the cosine of θ becomes negative and so this captures the intuition that as the angle becomes smaller and smaller X and I are more and more similar to each other and it is more likely that X will give a higher rating to item I so the way we make predictions is as follows given the user X we compute the cosine similarity between that user and all the items in the catalog and then you pick the items with the highest cosine similarity and recommend those to the user so that's the theory of content-based recommendations.

Now let's look at some of the pros and cons of the content-based recommendation approach.

Pros:

1. The main Benefit of the content-based recommendation strategy is that you don't require details on other users to create suggestions to a single user that turned out to be a very positive thing as you realize that you can start operating or creating content-based suggestions with a very first user from day one.
2. Another good thing about content-based recommendation is that you can recommend to users with very unique tastes when we go when you get to collaborate a filtering we see that collaborating collaborative filtering to make recommendations to a user we need to find other similar users the problem with that is that if there's a user with very unique or idiosyncratic tastes they may not be any other similar users whereas a content-based approach is able to deal naturally with this with the fact that you can make you know user can have very unique tastes as long as the if we can build item profiles for

the items that the user likes and a user profile for the user based on that we can make recommendations to that user A.

3. Third row is that variable to recommend new and unpopular items now when a new item comes in we don't need any ratings from users to build the item profile the item profile depends entirely on the features of the items are not on how other users rated the item so we don't have a so-called first grader problem that we will see in the in the collaborative filtering approach we can make recommendations for an item as soon as it becomes available and finally whenever the content based approach makes a recommendation you can provide an explanation or to the user for why a certain item was recommended in particular you can just list the content features that cost the item to be recommended for example if you recommend a news article to use of example using a cotton bait approach you may be able to say look in the past you spent a lot of time reading article that mention in Syria and that's why I am recommending this article on Syria to you.

Cons:

1. The most serious problem with the content-based approach either finding the appropriate features is very very hard for example how do you find features for images or movies producing now in the case of movies we suggested a set of features that include actors and directors and so on but it turns out that movies often cross shonduras and users are not very often loyal to specific actors or directors and the similar case of music it's very hard to sort of you know box music into specific genres and musicians and so on and images of course you know the features are very very hard to find so in general the finding appropriate features to make content based approach this work turns out to be a very very hard problem

and this is the main reason why the content-based approach is not more popular .

2. The second problem is one of over specialization remember the user profile is built using the item profiles of the the items that the user has rated or purchased now because of this if a user has never rated a certain kind of movie or a certain genre of movie he will never be recommended a movie in that in that genre for example or he'll never be recommended a piece of music that's outside his previous preferences in general people might have multiple interests and might express only some of them in the past and so it's hard to you know so it's very easy this way to miss recommending interesting items users because you don't have any fun enough data on the user.

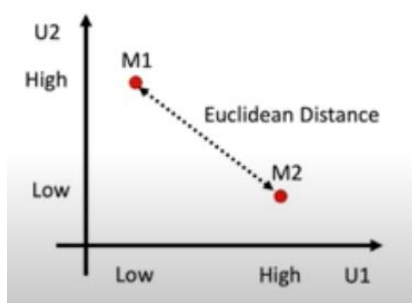
3. Another serious problem of the content-based approach is that it's unable to exploit the quality judgments of other users for example there might be a certain video or a movie that's widely popular across the you know wide cross-section of users however the current user has not expressed interest in that kind of movie and therefore the content-based approach will never recommend that movie to that user the final problem that we have the cotton based approach is one of a cold start problem for new users remember the user profile is built by aggregating item profiles of the items the user has rated when you have a new user the new user has not related any items and so the source so there is no user profile so there's a challenging problem of how to build a user profile for a new user in most practical situations new users start with you know most recommender system start of new users with some kind of average profile based on a system-wide average and then over time the user profile evolves as user rates more and more items and becomes more individualized to the use.

2. Collaborative Filtering

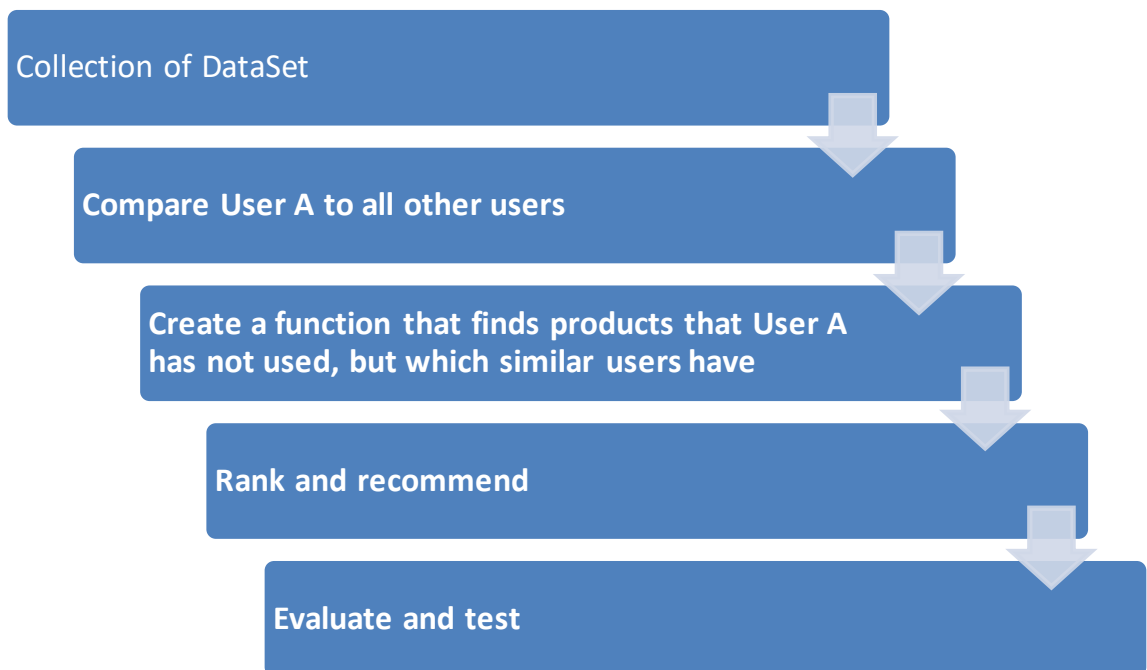
	User 1	User 2	User 3
Movie 1	5	2	4
Movie 2	3	4	3
Movie 3	1	4	1
Movie 4	2		??
Movie 5	5	2	??

Let's presume we've been provided a series of ratings from the past and our aim is to suggest films to this user 3 then how can we tackle this question well the first path we should take is to look at these ratings to seek to locate certain users that have displayed similar behaviour to our user 3 in this situation, let's try to find a user that seems more identical to user 3. While user two seems to like it, it's obvious that user 1 is more similar to user 3 now that we know this let 's find other movies that user 1 lights but user 3 hasn't watched it looks like movie Phi is the one the user liked and now we can recommend this movie to user 3 as well as this first approach where we're trying to find similar users and then recommend movies to user 3 Similar to it based on the ratings given by the other users in this case we know that user 3 likes film 1 so let's try to find similar films for film 1 and again we see that Mui Phi is the one where other users have also ranked it similarly so that this approach where we find similar items based on the items themselves is called as an item to collaborative filtering in practice what we know You might like it to grow older in your early teens, whereas in an item to item method the item remains the same irrespective of the time right, a horror film is still the horror film after 10 years, so let's do a simple quiz to make sure you get it based on the ratings given, so you can recommend movies to the user if you got it right and use it.

Quantifying the similarity



6.2 Procedure:



6.2.1 : Dataset

The data was collected from Movielens dataset. The snapshot of the dataset is:

The dataset consists of 1682 sample rows with with features `item_id`(Movies unique id), `title`(Movie name).

```
In [139]: movie_titles = pd.read_csv("Movie_Id_Titles")  
movie_titles.head()
```

```
Out[139]:
```

	item_id	title
0	1	Toy Story (1995)
1	2	GoldenEye (1995)
2	3	Four Rooms (1995)
3	4	Get Shorty (1995)
4	5	Copycat (1995)

And another table with user details consist of 100,003 sample rows with features `user_id`, `item_id`, `rating`, `timestamp`.

```
In [138]: df.head()
```

```
Out[138]:
```

	user_id	item_id	rating	timestamp
0	0	50	5	881250949
1	0	172	5	881250949
2	0	133	1	881250949
3	196	242	3	881250949
4	186	302	3	891717742

Now for further analysis we will merge both user and movie table together:


```
In [140]: df = pd.merge(df, movie_titles, on='item_id')
          df.head()
```

```
Out[140]:
```

	user_id	item_id	rating	timestamp	title
0	0	50	5	881250949	Star Wars (1977)
1	290	50	5	880473582	Star Wars (1977)
2	79	50	4	891271545	Star Wars (1977)
3	2	50	5	888552084	Star Wars (1977)
4	8	50	5	879362124	Star Wars (1977)

EDA

Now let's explore the data and get a look at some best rated movies.

Real data is often rather chaotic. Real data sets like bad formatting, trailing spaces, duplicates, empty rows, synonyms of different abbreviations, difference in scales, inconsistency in description, skewed distributions and outliers, and missing values are many possible issues.

Each of these issues can cause data analysis problems, and is worthy of attention in exploratory data analysis. There's substantial evidence of widespread data errors.

The error levels in clinical testing are estimated to be between 2.3% and 26.9%. A apparently insignificant error may also have severe repercussions. NASA lost a \$125 M Mars orbiter in 1999 because a team of engineers failed to convert measurements from English into metric units.

Data exploration often represents the first step in data analysis. It involves summing up the data set and making it real. Until some modeling research, it should be finished. As well as the negative effects of data errors as described earlier, large data may also trigger modern problems.

As an example, a modern approach can fail if a numerical variable encounters a long numerical value. This is a case of the growing study of yield regression. Analysis carried out on massive data may also suffer from the commonly

known effect of garbage in, garbage out. In other terms, data processing that is based on wrong data may also be deceptive.

Exploration of data serves several ends. First, it helps to gain insight over the data. This is sometimes called data comprehension. These insight may be very helpful in the study of subsequent results. And they could even dictate a suitable choice of modelling techniques or tools. Secondly, it helps to carry out data-sanitary checks to ensure a data Looks sensible, in the correct format and on the right scale.

Third, it helps to find out if any outliers of values are missing. Finally, summing up the data exploration data is common. These include an exploration of both the numerical summary and data. How should we conduct the data exploration? The simplest way is to scan the data manually.

You may have reviewed the data manually in the past for those of you who have experienced working with spreadsheets. However it is labor intensive and not feasible to manually review data. Even a spreadsheet to Excel can have as many as 1 million rows. There are several avenues to use large data collections. The first, is to look at data samples. Act for Data samples we can communicate explicitly with raw data also with large databases, which will significantly allow us to get a understanding of the results. However, in the end, we will either review the data using graphical description or confirmation of the results.

Visualization Imports

Using Matplotlib and seaborn Library present in python.

```
import matplotlib.pyplot as plt
import seaborn as sns
```

Let's create a ratings dataframe with average rating and number of ratings:

Using pandas library function groupby with movie title and the rating provided to that particular movie.

```
df.groupby('title')['rating'].mean().sort_values(ascending=False).head()
```

```
title
Marlene Dietrich: Shadow and Light (1996)    5.0
Prefontaine (1997)                          5.0
Santa with Muscles (1996)                   5.0
Star Kid (1997)                             5.0
Someone Else's America (1995)               5.0
Name: rating, dtype: float64
```

Now set the number of ratings column:

```
ratings['num of ratings'] = pd.DataFrame(df.groupby('title')['rating'].count())
ratings.head()
```

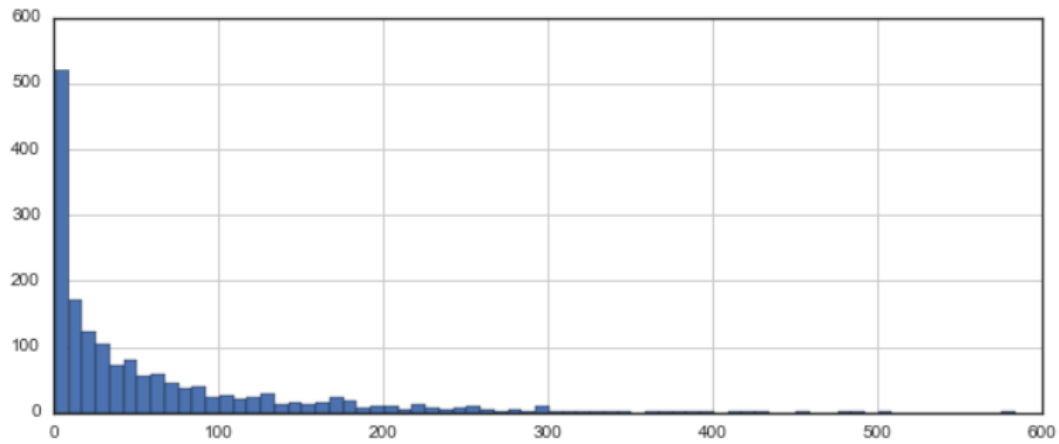
	rating	num of ratings
title		
'Til There Was You (1997)	2.333333	9
1-900 (1994)	2.600000	5
101 Dalmatians (1996)	2.908257	109
12 Angry Men (1957)	4.344000	125
187 (1997)	3.024390	41

Now a few histograms:

1. Number of ratings given by user.

```
plt.figure(figsize=(10,4))
ratings['num of ratings'].hist(bins=70)
```

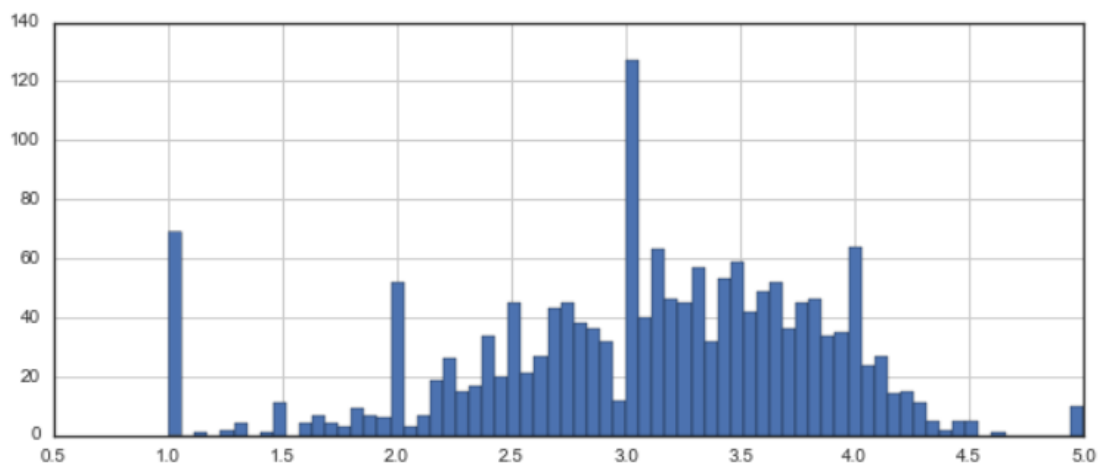
<matplotlib.axes._subplots.AxesSubplot at 0x1258f8780>



2. Ratings given by user to see the outliers.

```
plt.figure(figsize=(10,4))
ratings['rating'].hist(bins=70)
```

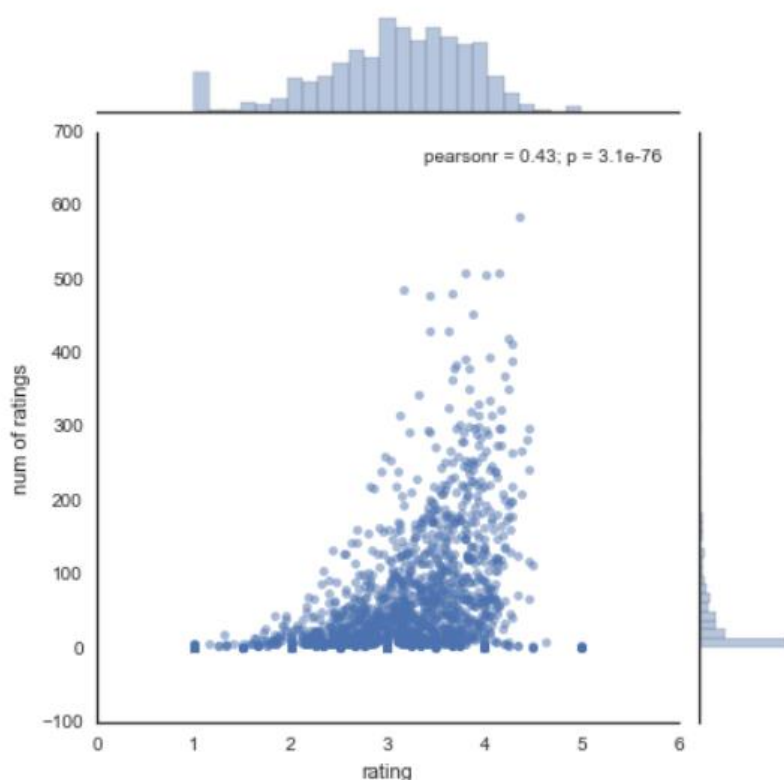
<matplotlib.axes._subplots.AxesSubplot at 0x125d12908>



Now Visualizing Using seaborn library.

```
sns.jointplot(x='rating',y='num of ratings',data=ratings,alpha=0.5)
```

```
<seaborn.axisgrid.JointGrid at 0x126005320>
```



Okay! Now that we have a general idea of what the data looks like, let's move on to creating a simple recommendation system.

Where the dots are dense is the place where most ratings have took place.

Recommending Similar Movies

Now let's create a matrix that has the user ids on one axis and the movie title on another axis. Each cell will then consist of the rating the user gave to that movie. Note there will be a lot of NaN values, because most people have not seen most of the movies.

```
In [149]: moviemat = df.pivot_table(index='user_id',columns='title',values='rating')
moviemat.head()
```

```
Out[149]:
```

	'Til There Was You (1997)	1-800 (1994)	Dalmatians (1996)	101 Angry Men (1957)	12 (1997)	187 (1996)	2 Days in the Valley (1996)	20,000 Leagues Under the Sea (1954)	2001: A Space Odyssey (1968)	3 Ninjas: High Noon At Mega Mountain (1998)	39 Steps, The (1935)	... (1994)	Yankee Zulu (1994)	Year of the Horse (1997)	You So Crazy (1994)	Frankenstein (1974)	Young Guns (1988)	Young Guns II (1990)	Poi: Han The
user_id	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	2.0	5.0	NaN	NaN	3.0	4.0	NaN	NaN	...	NaN	NaN	NaN	NaN	5.0	3.0	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.0	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	2.0	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN

```
5 rows x 1864 columns
```

Let's choose two movies: starwars, a sci-fi movie. And Liar Liar, a comedy.

```
: starwars_user_ratings = moviemat['Star Wars (1977)']
liarliar_user_ratings = moviemat['Liar Liar (1997)']
starwars_user_ratings.head()
```

We can then use `corrwith()` method to get correlations between two pandas series.

```
: similar_to_starwars = moviemat.corrwith(starwars_user_ratings)
similar_to_liarliar = moviemat.corrwith(liarliar_user_ratings)
```

Let's clean this by removing NaN values and using a DataFrame instead of a series.

```
corr_starwars = pd.DataFrame(similar_to_starwars, columns=['Correlation'])
corr_starwars.dropna(inplace=True)
corr_starwars.head()
```

	Correlation
title	
'Til There Was You (1997)	0.872872
1-900 (1994)	-0.645497
101 Dalmatians (1996)	0.211132
12 Angry Men (1957)	0.184289
187 (1997)	0.027398

Now if we sort the dataframe by correlation, we should get the most similar movies, however note that we get some results that don't really make sense. This is because there are a lot of movies only watched once by users who also watched star wars (it was the most popular movie).

```
corr_starwars.sort_values('Correlation', ascending=False).head(10)
```

	Correlation
title	
Commandments (1997)	1.0
Cosi (1996)	1.0
No Escape (1994)	1.0
Stripes (1981)	1.0
Man of the Year (1995)	1.0
Hollow Reed (1996)	1.0
Beans of Egypt, Maine, The (1994)	1.0
Good Man in Africa, A (1994)	1.0
Old Lady Who Walked in the Sea, The (Vieille qui marchait dans la mer, La) (1991)	1.0
Outlaw, The (1943)	1.0

Let's fix this by filtering out movies that have less than 100 reviews (this value was chosen based off the histogram from earlier).

Now sort the values and notice how the titles make a lot more sense.

```
corr_starwars[corr_starwars['num of ratings']>100].sort_values('Correlation',ascending=False).head()
```

	Correlation	num of ratings
title		
Star Wars (1977)	1.000000	584
Empire Strikes Back, The (1980)	0.748353	368
Return of the Jedi (1983)	0.672556	507
Raiders of the Lost Ark (1981)	0.536117	420
Austin Powers: International Man of Mystery (1997)	0.377433	130

Now these are the movies that are more correlated with with star wars so these are the movies that will be recommended to the user watching star wars.

Same will be done for Lair Lair comedy movie.

References

- <https://movielens.org/>
- <https://www.udemy.com/>
- www.python.org

Abdul Hanan et al. Emotion Detection of Text, International Journal of Engineering Research and Development e-ISSN: 2278-067X, p-ISSN: 2278-800X, www.ijerd.com Volume 11, Issue 07 (July 2015), PP.23-34

am2

ORIGINALITY REPORT

18%	7%	7%	11%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to University of Melbourne Student Paper	2%
2	Submitted to Griffith College Dublin Student Paper	2%
3	Submitted to CITY College, Affiliated Institute of the University of Sheffield Student Paper	1%
4	ccsenet.org Internet Source	1%
5	"Smart Technologies and Innovation for a Sustainable Future", Springer Science and Business Media LLC, 2019 Publication	1%
6	Bei-Bei Cui. "Design and Implementation of Movie Recommendation System Based on Knn Collaborative Filtering Algorithm", ITM Web of Conferences, 2017 Publication	1%
7	fsegitlab.wlv.ac.uk Internet Source	1%